UNITED STATES DISTRICT COURT

NORTHERN DISTRICT OF CALIFORNIA

SAN FRANCISCO DIVISION

NETWORK APPLIANCE, INC.,

Plaintiff-Counterclaim Defendant,

v.

SUN MICROSYSTEMS, INC.

Defendant-Counterclaim Plaintiff.

CASE NO. C-07-06053-EDL

DECLARATION OF DR. DONALD ALPERT IN SUPPORT OF SUN MICROSYSTEMS INC.'S RESPONSIVE CLAIM CONSTRUCTION BRIEF

Claim Construction Hearing:
Date: August 27, 2008
Time: 9:00 a.m.
Judge: Hon. Elizabeth D. Laporte

I, Dr. Donald Alpert, declare as follows:

1. I am an independent consultant with Camelback Computer Architecture, LLC. My residence and place of business is at 5820 N. 46th Place, Phoenix, Arizona. I am over the age of eighteen, and I am a citizen of the United States. I have been engaged by Sun Microsystems, Inc. to provide my opinions regarding the technology of U.S. Patent No. 5,749,095 (the '095 patent).

2. My declaration will provide a summary of my experience, a list of the materials I reviewed, and a description of some aspects of computer systems that will assist in understanding the inventions described in one of the patents-in-suit. Additionally, I discuss the meaning of claims 1, 11 and 17 of the '095 patent. Specifically, I consider the meaning of the phrase "completing [a] write operation within [a] local processing node …" as recited in claim 1 and of the phrase "complete [a] write operation with respect to [a] processor…" as recited in claims 11 and 17. Herein, I provide my opinion as to how one of ordinary skill in the art of computer systems and architecture would have understood this term in these claims at the time of invention.

I. EXPERTISE

3. I have over 30 years of academic and industrial experience in applying, designing, studying, teaching, and writing about microprocessors and computer systems. I received an Electrical Engineering PhD degree in 1984 from Stanford University. I earlier received an Electrical Engineering BS degree from MIT in 1973 and an Electrical Engineering MS degree from Stanford University in 1978. I have taught classes in computer architecture at Stanford, Tel Aviv, and Arizona State Universities.

4. I was the lead architect for the design of three high-performance microprocessors at Zilog and National Semiconductor from 1980 to 1989. Later at Intel, I was the lead architect of the Pentium® Processor from 1989 to 1992 and of the 815 integrated-graphics chipset from 1999 to 2000.

5. I am a Senior Member of the Institute of Electrical and Electronics Engineers (IEEE), and served as the chair of the IEEE Technical Committee on Microprocessors and Microcomputers from 1999 to 2000. I served as Session Chair for Cache Design at the IEEE International Symposium on Computer Architecture (1989). I was the keynote speaker at the first Cool Chips conference (1998), dedicated to the study of low-power microprocessors and systems. I have given invited lectures at several universities, and published ten papers on computer architecture in various professional journals and conference proceedings. My paper entitled “Architecture of the Pentium Processor” was selected as best paper in IEEE Micro for 1993. I am a named inventor on over 30 U.S. patents that pertain to microprocessors and related technology.

6. My *curriculum vitae*, including a list of all of my publications, and cases in which I have previously testified as an expert is attached as Exhibit A.

7. I make this declaration in support of Sun’s proposed claim construction related to the ’095 patent. I make this declaration of my own personal knowledge, and if called as a witness, I could and would testify competently to the facts in this declaration.

II. MATERIALS REVIEWED

8. I have studied U.S. Patent No. 5,749,095, issued May 5, 1998, naming Erik E.

Hagersten as the inventor and entitled "Multiprocessing System Configured to Perform Efficient Write Operations." I reviewed the file including U.S. Application Serial No. 675,634, filed July 1, 1996 related to the ’095 Patent. Additionally, I reviewed each of the documents cited below.

III. LEVEL OF SKILL IN THE ART

9. Based on my reading of the patent specification and claims, I believe that one of ordinary skill in the art at the time of the invention of the ’095 patent would have had a master’s degree in Electrical Engineering, Computer Science, or related discipline, as well as extensive knowledge of shared memory multiprocessor systems, including technologies of cache memory, cache coherence, and interconnection networks. Alternatively, a person of ordinary skill in the art can be a person with 2 or more years practical equivalent experience with Distributed Shared Memory multiprocessor computer systems.

IV. TECHNICAL BACKGROUND PERTINENT TO THE ’095 PATENT

10. The summary of technology background below is based on my general knowledge with relevant citations to the ‘095 patent.

11. The ’095 patent relates to the field of computer systems and the methods available within computer systems for data to be reliably stored in memory. [’095 patent, col. 3:5-8] The ‘095 patent improves computer system performance by teaching a novel technique for storing modified data in a memory shared by multiple processors. [‘095 patent, Abstract]

1. Single-Processor Computers

12. A basic computer system is comprised of a central processing unit (CPU or processor) and a main memory device, which stores data values at numerous locations. When a processor reads or writes a data value in main memory, the processor specifies the data value’s memory location by its memory "address." [3:33-55, 8:37-39, 10:4-6]

13. Modern processors typically also contain a relatively small memory known as a "cache." [3:33-36] Processors combine the use of a small, fast cache with larger, slower main memory.

14. This use of cache memory introduces some complications. By storing data in

main memory and a cache memory, two copies of a single data item exist in the system: One copy would be stored in the cache memory, and another copy of the data, which was potentially outdated or "stale," would be stored in the main memory. If another device, such as a disk drive, were then to read the stale data from main memory, this could result in a catastrophic system crash, or often worse, an undetected computing error. Thus, when a processor or other device modifies a memory location, it is common for computer systems to avoid such problems by invalidating or updating a copy of the location. [3:43-50] The cache keeps a status for the data, such as whether the copy is valid or modified, in order to implement correct and efficient handling of invalidations and updates. For example, when a processor writes to a location with modified status, it is unnecessary for the processor to communicate invalidations or update to other system components.

2. Multiple-Processor Computers with Shared Memory

15. The use of computer systems to handle increasingly complex computing tasks has led to using two or more processors in a computer system. When the processors share at least a portion of memory, the system is called a "multiprocessor." In a multiprocessor, one processor can be executing one program and accessing data in memory at the same time that another processor is executing a second computer program and accessing data in memory for that computer program. Also, multiple processors may be executing different parts of a single computer program to help complete the task of the computer program more quickly. [3:11-21] When any processor can (1) access any memory location and (2) execute any program, the system organization is known as a Symmetric MultiProcessor (SMP).

16. The problem of stale memory data, described above for single processor systems, becomes more complex for multiprocessors because a copy of a memory location may exist within the cache of each processor. Nevertheless, software is commonly written with the expectation that the value read from a memory location is the same, regardless of which processor is performing the access, a characteristic known as "cache coherence." Thus, when a processor or other device modifies a memory location, potential problems of stale data should be avoided by invalidating or updating all copies of the location. [3:36-50]

17. One way of interconnecting multiple processors and memory is using a single, shared bus. Each processor accesses all of memory using the bus as a common path, and thus the time for any processor to access any memory location is approximately constant. Consequently, such an organization is called “Uniform Memory Architecture” (UMA). [3:22-32] The problem of stale data arising in the caches of processors interconnected by a single bus is generally avoided by use of bus snooping protocols, where each cache observes (“snoops”) the memory accesses by other caches communicated over the bus. By listening to the transactions on the shared bus, each cache can determine whether to update or invalidate a copy of the addressed memory location. [3:50-55]

18. As the speed and number of processors using a shared bus increases, it becomes increasingly difficult and expensive to increase the bus’s performance correspondingly. A bus is inherently limited in its scalability by the number of signals, switching speed, and transmission line characteristics. [4:1-18]

3. Multiple-Processor Computers with Distributed Shared Memory.

19. The bottleneck of a single shared bus can be overcome by distributing the main memory along with each processor of the multiprocessor system, then using a network to interconnect the processors and associated memories. Compared with a single, shared bus an interconnection network supports multiple, simultaneous communication paths with faster, point-to-point signaling. Each processor and associated memory comprises a “node.” [4:19-23]

20. An example distributed shared memory (DSM) system with eight processors is reproduced below from Figure 1A of the ‘095 patent.

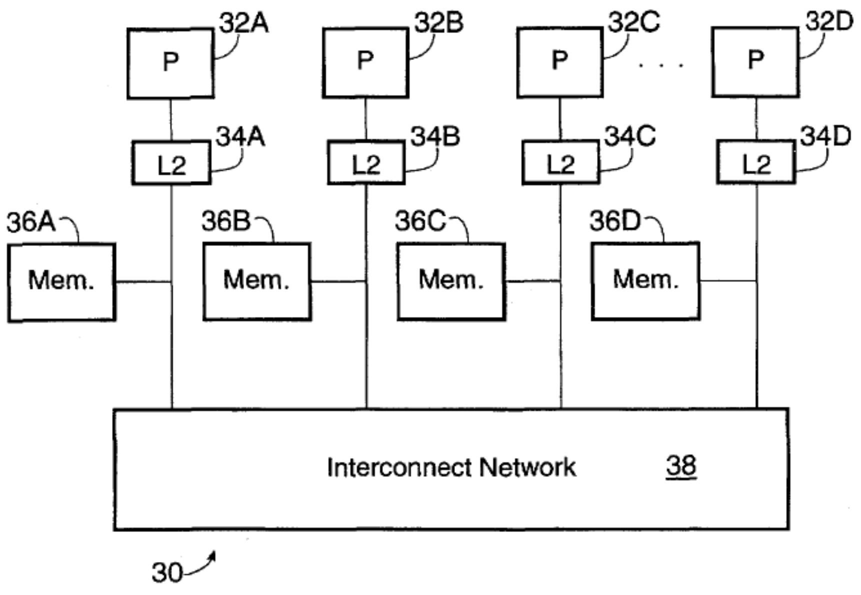


Fig. 1A

21. An advantage of this multiprocessor system is that the design is scalable. That is, if it is desirable to increase performance, adding more nodes will increase the number of processors, memory, and communication bandwidth correspondingly. Hence, the system is highly expandable in a balanced manner. [4:31-39] But this form of convenient expansion comes with a disadvantage: Because main memory is now distributed among nodes and not every node has immediate access to every other node, the time it takes to perform a memory access is dependent on the distance through the network between the node making the request and the node that holds the data. [4:41-44] Because different memory accesses take varying amounts of time, such a system is called a nonuniform memory architecture (NUMA). [10:7-24]

22. The problem of stale memory data becomes more complicated for DSM than with a single shared bus. When a processor or other device modifies a memory location, all copies of data are to be updated or invalidated. One way to accomplish this is to broadcast a message to all nodes throughout the network, but that causes an excessive amount of memory traffic. Furthermore, most of that traffic would be wasteful because it is inherently inefficient, and therefore an uncommon occurrence, to have many copies of data that are being modified located in caches throughout the system.

23. The solution taught by the ‘095 patent is to instead keep track of which other

nodes have cached a copy of the location in the node that contains the main memory location, which is called the location's "home" node. This information is held in a table, called a "directory," associated with the memory. [4:26-29]

V. THE '095 PATENT EMBODIMENTS

24. The '095 patent describes a system (10) that comprises multiple DSM nodes (SMP Nodes 12A-D) interconnected by a network (14). Within a node (12A), a single bus (SMP Bus 20) interconnects one or more processors (16A-B) with associated caches (18A-B), memory (22), I/O interface (25), and system interface (24). [8:50-9:53] Figure 2 shows an example node implementation with four processors, but a node can have a single processor or any number of processors. [9:54-56, 11:18-29, 31:17-20]

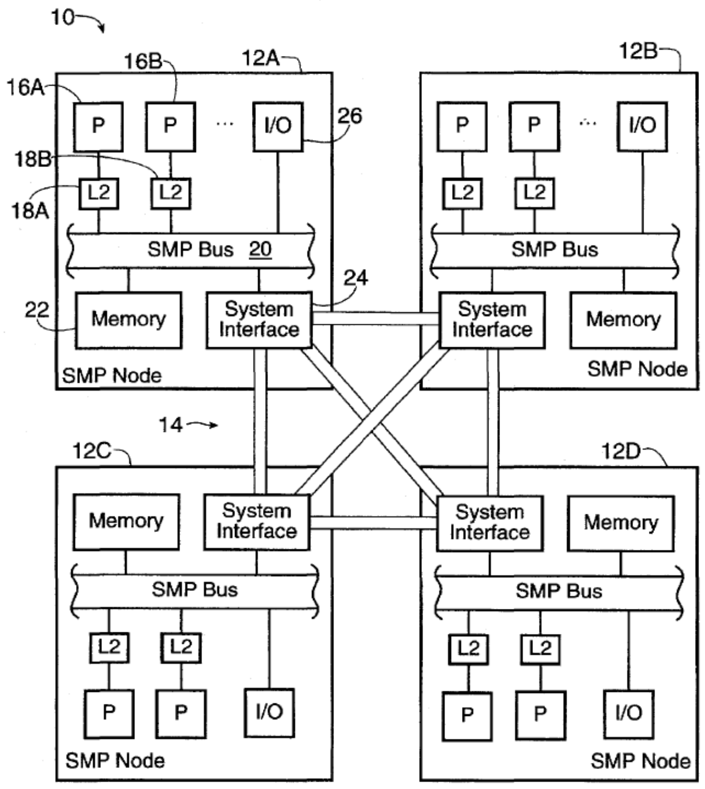


Fig. 1

25. Cache coherence within each SMP node is maintained using snooping. Cache coherence is maintained between SMP nodes using the system interface (24), network (14), and directories associated with each memory (22) which are kept in the home node [9:34-39] An

embodiment of the ‘095 invention supports two types of distributed memory architecture. [9:58-60, 11:1-10] One type of architecture is NUMA, as described above, for which “[e]ach address within the address space corresponds to a location within one of memories 36 [22].” [10:5-6, see also 10:62-64] Another type of architecture is Cache-Only Memory Architecture (COMA). COMA has the characteristics described at 10:58-62:

> No permanent mapping of a particular address to a particular storage location is assigned. Instead, the location storing data corresponding to the particular address changes dynamically based upon the processors 42 which access that particular address.

The description below focuses on NUMA because operation for that architecture is simpler than for COMA, and the detailed differences between NUMA and COMA are not important to the meaning of the claim terms.

26. A processor on one node (a “local node” for the processor) can write to a memory location on another node (a “remote node” for the processor, the “home node” for the memory location). [31:14-35] When no copy of the location is present in caches of the local node, the local node must communicate with the remote node and potentially other nodes in order to perform a coherency operation to ensure cache coherence. An example protocol for such communication is shown in Figure 4 (reproduced below) and described at 18:1-19:12.

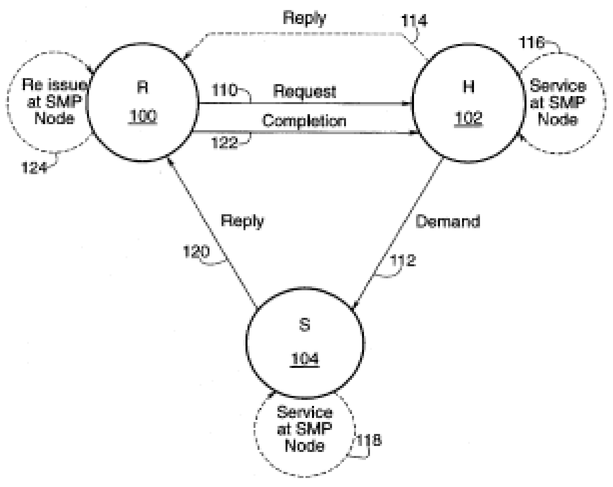


Fig.4

27. For an exemplary coherency operation, the local node (Request Agent 100) first transmits a coherency request (122) to the Home Agent (102). The home agent sends a coherency demand (112) to any Slave Agent (104) that may have a copy of the memory location and a coherency reply (114) to the request agent, indicating the number of slave agents that will reply to the demand. Each slave agent that received a coherency demand sends a reply (120) to the request agent. When the request agent has received a coherency reply from each affected slave agent, requesting agent sends a coherency completion (122) to the home agent. The time to complete communication through the network for the coherency operation may be a few microseconds; in contrast, a processor's access to memory within a local node may take only a few hundred nanoseconds. [10:23-25]

28. The '095 patent explains that the latency to perform coherency operations for remote accesses can be a severe problem for prior-art DSM systems:

> Unfortunately, processor access to memory stored in a remote node (i.e. a node other than the node containing the processor) is significantly slower than access to memory within the node. In particular, write operations may suffer from severe performance degradation in a distributed shared memory system. If a write operation is performed by a processor in a particular node and the particular node does not have write permission to the coherency unit affected by the write operation, then the write operation is typically stalled until write permission is acquired from the remainder of the system. Stalling the write may occupy processor resources (such as storage locations for the write data) until the write permission is acquired. Accordingly, the processor resources are not available for use by subsequent operations, thus possibly further stalling processor execution. A more efficient method for performing write operations in a distributed shared memory system is desired. [4:41-57]

29. "Write stream" operations are a specific type of write operation that update an entire coherency unit, which contains numerous bytes of data. [8:4-5, 23:27-28] When a write stream operation is performed, it is unnecessary for the processor to read a copy of the coherency unit before performing the write because the entire area is written. For other types of write operations, where less than all bytes of a coherency unit are to be modified, the operation may have to read the coherency unit before modifying any data within the coherency unit. [8:5-7]

30. One application of write stream operations is copying large blocks of data from

one memory location to another. [8:18-32, 23:27-28] For example, an operating system may manage memory in units of pages, which occupy 4096 bytes, in a system with coherency units of 64B [13:13-40, 14:26-28, 15:3-4] When the operating system copies a page from one location to another, this involves writing 64 coherency units to memory.

31. Thus, when copying a page from a location within the local node to a location at a remote node, there will be a large number of write stream operations in close succession. In the prior art, as explained above, each such write stream operation may become stalled for several microseconds waiting for its associated coherency operation to complete. With only a limited number of buffers available to store write stream data, the resources of the processor may be exhausted and the processor itself will stall executing instructions when the buffers become fully occupied. [28:63-66, 29:6-20]

32. The ’095 patent teaches that under certain circumstances, a computer system may avoid stalling a write operation until its associated coherency operation is complete. That is, the ’095 patent recognizes that the coherency operations can overlap with or occur after the write operations in some cases. [5:5-8, 37:25-28, 31:36-41]

33. The inventions of the ‘095 patent in large part relates to novel techniques disclosed that allow certain write operations, known as “fast writes,” in DSM systems to complete before associated coherency operations have completed. These techniques provide advantages of higher performance.

34. The invention of the ‘095 patent teaches techniques that enable benefits of overlapping write operations with associated coherency operations while ensuring that the computer system functions correctly. More specifically, the patent discloses several forms of encodings that can distinguish between a “fast write operation,” that is a write operation that may be performed prior to or while acquiring write permission to the coherency unit, and other write operations. [4:62-65, 5:17-27] For example, a predefined encoding for fast write operations can be provided as part of the address or instruction. [Figure 14, 7:61-67, 26:43-27:14, 27:58-64]

35. When a large number of write stream operations occur in sequence, performance can be improved by using fast write operations. [5:10-12, 8:13-28] The small number of stream

buffers within the processor can be freed up quickly. [29:6-52] Performance comparable to the throughput of the SMP Bus can be achieved instead of being limited by the interconnect network latency. [31:7-13] Fast write operations can also provide performance improvement for other applications in addition to block copies using write-stream operations. [5:17-28, 8:13-19]

VI. UNDERSTANDING OF PERSON OF ORDINARY SKILL IN THE ART OF COMPLETING THE WRITE WITH RESPECT TO THE REQUESTING NODE

36. I am informed that terms in patent claims must be construed as a first step in analyzing whether a claim is infringed and/or whether the claim is invalid. I am further informed that patent claims must be construed the same way for infringement and invalidity analyses. I understand that claims are to be interpreted in light of the specification and file history, and that claim construction starts with the plain language of the claims.

37. It is my understanding that the ‘095 Patent claim terms should be interpreted using their plain and ordinary meaning to one of skill in the art at the time the earliest priority application was filed (July 1, 1996), unless the inventor acted as his or her own lexicographer and coined new terms or redefined a well-known term of art. I have applied the definition of ¶9 for a person skilled in the art. Because disputed terms have specific usage in the art, I have also included references to technical publications that would be known to a person skilled in the art.

38. In the paragraphs below I provide a summary of my analysis and my opinion for the meaning of the phrases “completing [a] write operation within [a] local processing node …” as recited in claim 1 and “complete [a] write operation with respect to [a] processor…” as recited in claims 11 and 17. The parties have proposed the constructions shown in the table below.

Claim Term	**NetApp’s proposed construction**	**Sun’s proposed construction**
“completing [a] write operation within [a] local processing node …” in claim 1	“transferring the write data from an initiating processor to a system interface.”	This phrase does not require construction because the phrase is clear on its face. However, if the Court decides to construe this phrase, the phrase should be construed to mean: “data for the write operation (1) is provided to subsequent read operations within the local processing node to the same address as the write operation and (2) is or will be coherent within distributed shared memory.”

"complete [a] write operation with respect to [a] processor …" in claims 11 and 17	"transferring the write data from an initiating processor to a system interface."	This phrase does not require construction because the phrase is clear on its face. However, if the Court decides to construe this phrase, the phrase should be construed to mean: "data for the write operation (1) is provided to subsequent read operations by a processor to the same address as the write operation and (2) is or will be coherent within distributed shared memory."

Understanding of one skilled in the art

39. The meaning of cache coherency, and more specifically of "completing a write," would be well known to one skilled when the '095 patent was filed. Persons of ordinary skill in the art would understand that completing the write operation involves having the data for the memory write be returned to subsequent memory reads and that the coherency operation is guaranteed to be performed.

40. Persons of ordinary skill in the art further understand that a write operation may complete at different points within a computer system. A write operation from the perspective of ordinary skill in the art is complete with respect to a certain portion of the computer system if that portion of the computer system will receive the write data during a subsequent read and is guaranteed that the coherency is or will be complete within the entire system. For example, if a write operation is complete with respect to a node in the system, the node can read the data of the write operation from the same address as the write operation and the coherency is guaranteed to be completed.

41. The understanding of "completion" of a write within the art is associated with the ability to perform a subsequent read operation. Such an interpretation is supported by the '095 specification, for example at 3:41-50:

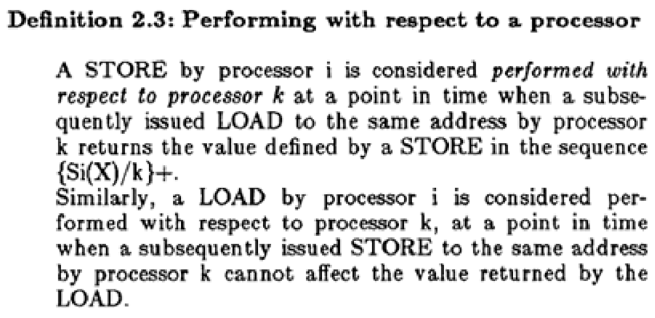
> Generally speaking, an operation is coherent if the effects of the operation upon data stored at a particular memory address are reflected in each copy of the data within the cache hierarchy. For example, when data stored at a particular memory address is updated, the update may be supplied to the caches which are storing copies of the previous data. Alternatively, the copies of the previous data may be invalidated in the caches such that a subsequent access to the particular memory address causes the updated copy to be transferred from main memory.

42. Such an interpretation is further supported by the specification of related United States Patent 5,829,033 attached as Exhibit B, which was incorporated by reference into the ‘095 patent as application Ser. No. 08/673,059 at 1:6-12. For example, this incorporated reference states at 1:60-2:2

> Understandably, cache coherence is important. If multiple devices have local copies of the same memory location in their caches, correct system operation dictates that all devices must observe the same data in their caches (since they are meant to hold copies of the same memory location). But if one or more devices write to their local copy of the data in their caches, all devices may no longer observe the same data. Cache coherence is the task of ensuring that all devices observe the same data in their caches for the same memory location.

43. Such an interpretation is also supported by a paper[1] published by Lucien N Censier and Paul Feautrier at p. 1113: “A memory scheme is coherent if the value returned on a LOAD instruction is always the value given by the latest STORE instruction with the same address.”

44. In addition, the paper[2] “*Correct memory operation of cache-based multiprocessors*” by Dr. Michel Dubois teaches the meaning of completing a write operation. More specifically, the paper defines *“performing [a STORE] with respect to a processor”* at 236:


Definition 2.3: Performing with respect to a processor

A STORE by processor i is considered *performed with respect to processor k* at a point in time when a subsequently issued LOAD to the same address by processor k returns the value defined by a STORE in the sequence {Si(X)/k}+.
Similarly, a LOAD by processor i is considered performed with respect to processor k, at a point in time when a subsequently issued STORE to the same address by processor k cannot affect the value returned by the LOAD.

45. One skilled in the art would understand that “performing a store” of Dubois’s paper has the same meaning as “completing a write” of the ‘095 patent, at least because the plain

---

[1] Censier, L. M. and Feautrier, P. 1978. A New Solution to Coherence Problems in Multicache Systems. IEEE Trans. Comput. 27, 12 (Dec. 1978), 1112-1118, attached as Exhibit C.

[2] Scheurich, C. and Dubois, M. 1987. Correct memory operation of cache-based multiprocessors. In *Proceedings of the 14th Annual international Symposium on Computer Architecture* , ISCA ’87, ACM, New York, NY, 234-243, attached as Exhibit D.

meanings of “perform” and “complete” are synonymous.[3] This supports the claim construction that “complete [a] write operation with respect to [a] processor” means “data for the write operation is provided to subsequent read operations by a processor to the same address as the write operation.”

46. Dubois’s paper further explains at p. 236: “In some architectures, for example, at time *t* a STORE may have been performed with respect to processor *x* but not yet with respect to processor *y*.” Thus, there is a clear distinction between a STORE being performed with respect to one processor or to all processors of a node. In the context of the ‘095 patent, one skilled in the art would understand that completing a write operation within a node would include completing the write operation with respect to all processors of the node. One skilled in the art would understand that the meaning of completing a write operation explained above also requires the write operation has been or will be committed to guarantee the completion of the coherency operation. That is, the written data is or will be coherent within distributed shared memory of the system.

NetApp’s proposed construction is flawed

47. Netapp’s proposed construction suffers from at least two serious flaws related to differentiating claims and claim terms. Firstly, NetApp proposes the same interpretation for two distinct claim terms. The claim terms distinguish between completing a write operation (1) with respect to a processor and (2) within a processing node. The distinct use and meaning of these terms is further supported by the specification, for example at 5:39-65. The construction must distinguish between the two claim terms; consequently Netapp’s single proposed construction cannot be correct for both disputed claim terms.

48. Furthermore, in Netapp’s proposal, the disputed terms require that the data be transferred from an initiating processor. Such a requirement is explicit in claim 9, which depends on claim 1: “9. The method as recited in claim 1 wherein said completing comprises transferring data from said processor.” [32:31-32] Therefore, transferring the data cannot be required to

[3] Merriam-Webster's Collegiate Thesaurus (http://unabridged.merriam-webster.com/cgi-bin/thesaurus?book=Thesaurus&va=perform)

complete the write operation within a local processing node because then claims 1 and 9 would be indistinguishable. Similarly, claim 11 must be distinguishable from claim 16, and therefore transferring the data cannot be required to complete the write operation with respect to a processor. Consequently, Netapp's proposed construction cannot be correct for either disputed claim term.

49. Moreover, one skilled in the art would understand that Netapp's proposed interpretation does not reflect the operation of embodiments disclosed by the '095 patent. For example, applying Netapp's construction to claim 1 would require "transferring the write data from an initiating processor to a system interface" subsequent to completing the coherency operation for a write operations that is not a fast write. [32:7-10]. Contrary to such a claim interpretation, embodiments of the '095 patent can complete a write operation without "transferring the write data from an initiating processor to a system interface," as explained below.

50. When a processor performs an ordinary write to a cacheable location in a remote node, if a cache miss occurs then there is a "Read to Own" (RTO) request for the missing location. [23:23-25] The coherency activity for this case is shown in Figure 5 and described at 19:27-20:27. After the coherency operation is complete, the missing coherency unit has been transferred to the requesting processor's cache and is in the modified state. [19:45-49] The processor then updates the cache to complete the write operation without transferring data to the system interface. Thus, Netapp's proposed construction is in contradiction to the disclosed embodiment.

51. In addition, Netapp's construction is inappropriate because it attempts to read a specific feature of one embodiment into the scope of the claims. For example, the specification discloses an association between transferring data from the processor to the system interface and completing a write, but this is in a context that explicitly is describing only one of many embodiments and the system interface [27:65-67, 28:49-52]. One skilled in the art would appreciate that the detailed description of one embodiment is useful to understand how to practice the invention, but that many modifications and alternative embodiments of the invention are

expected [7:8-17].

52. The '095 patent provides a Summary of the Invention section [4:60 –6:12]. The Summary section provides general and broad disclosure of the fast write method contemplated in the '095 patent [5:29-51]. This description is not limited to any hardware implementation. The paragraph recited at col. 5:39-51 loosely tracks the language of claim 1 [5:39-51]. The paragraph recited at col. 5:52-65 tracks the language of claim 11 [5:52-65]. The paragraph recited at col. 5:66 through col. 6:12 tracks the language of claim 17 [5:66 – 6:12].

53. When viewed from the perspective of ordinary skill in the art, the fast write protocol disclosed in the '095 patent is not tied to any hardware embodiment. The disclosure of the system interface within Figure 16 is one exemplary hardware platform on which the fast write protocol may be implemented [Fig. 16].

Sun's proposed construction is correct

54. As explained above, one skilled in the art would understand the plain meaning that completing a write in a DSM computer system relates to providing the data to subsequent reads by processors to the same memory address and to guaranteeing the completion of the coherency operation. Therefore, I agree with Sun's position that no construction of the disputed terms is necessary. Nevertheless, if the Court decides to construe the disputed terms, then the correct constructions are

- "completing [a] write operation within [a] local processing node" means "data for the write operation (1) is provided to subsequent read operations within the local processing node to the same address as the write operation and (2) is or will be coherent within distributed shared memory."
- "complete [a] write operation with respect to [a] processor" means "data for the write operation (1) is provided to subsequent read operations by a processor to the same address as the write operation and (2) is or will be coherent within distributed shared memory."

I declare under the penalty of perjury that the foregoing is true and correct. This declaration is executed on this 21st day of July, 2008, at Phoenix, Arizona.

Donald Alpert

DR. DONALD ALPERT